APPENDIX A

MPEG-4 Video Verification Model:
A video encoding/decoding algorithm based on content representation

Touradj Ebrahimi
Signal Processing Laboratory
Swiss Federal Institute of Technology - EPFL
CH-1015 Lausanne
Switzerland

Abstract

MPEG-4 video aims at providing standardized core technologies allowing efficient storage, transmission and manipulation of video data in multimedia environments. This is a challenging task given the broad spectrum of requirements and applications in multimedia. In order to achieve this broad goal, rather than a solution for a narrow set of applications, functionalities common to clusters of applications are under consideration. Therefore, video group activities in MPEG-4 aim at providing solutions in the form of tools and algorithms enabling functionalities such as efficient compression, object scalability, spatial and temporal scalability, and error resilience. The standardized MPEG-4 video will provide a toolbox containing tools and algorithms bringing solutions to the above mentioned functionalities and more.

The current focus of the MPEG-4 video group is the development of the Video Verification Models. A Verification Model (VM) is a common platform with a precise definition of encoding and decoding algorithms which can be presented as tools addressing specific functionalities. It evolves through time by means of core experiments. New algorithms/tools are added to the VM and old algorithms/tools are replaced in the VM by successful core experiments. So far, the MPEG-4 video group has focused its efforts on a single VM which has gradually evolved from version 1.0 to version 4.0, and in the process has addressed increasing number of desired functionalities, namely, content based object and temporal scalabilities, spatial scalability, error resilience, and compression efficiency.

This paper gives an overview of version 4.0 of the video VM in MPEG-4. In doing so, issues, algorithms, and majors tools used in the development of this future video standard are discussed.

1 INTRODUCTION

The Motion Picture Expert Group (MPEG) has successfully introduced two standards for coding of audiovisual information, known by acronyms MPEG-1 and MPEG-2. The first addresses the storage of audiovisual information on CD-ROMs [1], and the second handles the generic coding of digital TV and HDTV signals [2]. Both these standards have had tremendous impact on the consumer electronics industry and their use in a number of today's applications is a witness of this fact.

The consumer electronics industry, cable operators, broadcasters, telecommunication companies, computer and software firms have developed a growing interest in a new form of communication called *multimedia*. This trend was accelerated by the increasing use of CD-ROMs, the World Wide Web on Internet, and what is commonly called the *information superhighway*. What is multimedia and how to implement it? Very likely, each of the above players would give you a different definition and a different strategy to establish multimedia communications as an extension of their current core business. However, they would all agree that multimedia, whatever its definition, can be seen as a platform for exchange of information coming from different sources of possibly different nature. On this account, it will support a very broad spectrum of applications. Due to their inherent rigidity, current standards cannot adequately address the new expectations and requirements that arise from such diverse applications.

The variety of applications makes the audiovisual data representation a challenging problem, especially since most applications claiming to be multimedia possess the common feature of having some sort of interactivity with the data. Applications impose sets of specifications which may greatly vary from one to another. Diversity of applications implies diversity of collections of specifications. Each application is characterized by: the type of data to be processed (still pictures, video, stereo images, etc.), the nature of data (natural, synthetic, text, medical, graphics, etc.), the targeted bitrate (low, medium, high), the maximum admissible delay (ranging from real time to off-line), the type of communication (point to point, point to multi-point, multi-point to multi-point), and the set of functionalities offered (scalability, object manipulation, progressive transmission, editing, etc.).

The new work item known as MPEG-4 aims at providing a standard in order to cope with the requirements of current and future multimedia applications. The MPEG-4 activity started in July 1993 as a pro-active action in order to propose a solution to challenges in multimedia communications. A Working Draft of the standard is projected for November 1996 which will become an International Standard by November 1998.

This paper describes the verification model of the video part of the MPEG-4 standard, in its current version [3]. The paper is structured as follows. This section gives an overview of the video standardization process in MPEG-4 and addresses the functionalities currently under consideration. It also defines the concept of Verification Model (VM) and its evolution through core experiments. Section 2 discusses the architecture of the video verification model, its data structure, and the specific coding technique used for coding video content in form of occurrences of video objects in time. Section 3 presents an overview of coding tools currently offered by the

video verification model, namely, shape coding, motion estimation and compensation, texture coding, scalable coding, and error robustness. Section 4 concludes the paper.

1.1 General overview of video standardization process in MPEG-4

The MPEG committee has set up a process in order to provide an efficient method to reach adequate standards for audio-visual data communications. The goal of this section is not to analyze or describe this process in detail, as a number of publications already give a good description of this process [4, 5, 6]. However, a brief review of some key points is useful in understanding the current activities for the definition of the video VM.

The standardization process for MPEG-1, MPEG-2, and MPEG-4 can be divided in a number of steps [4], not necessarily in a successive order. These are: *requirements, competitive phase, selection of basic methods, collaborative phase, working draft and draft international standard, validation* and finally the *international standard.*

As far as the video part of the MPEG-4 is concerned, these steps have been implemented in the following way. A study of the scope and requirements in MPEG-4 video started as early as in July 1993. After about two years of investigation, a series of seminars, and other related activities to define MPEG-4 video issues, it became clear that the core technology of a number of multimedia applications need two key components that are not explicitly present in any of the previous standards. The first component is to base the technology on a content representation as opposed to a pixel or frame based representation, to allow interactivity with content. The second is to provide a certain degree of flexibility in the design of the final system so as to lead to an open yet efficient standard. To this end, the notion of toolbox, already present in MPEG-2, was extended to allow a flexible design of coding algorithms using tools selected based on the requirements of specific applications.

For higher efficiency and in order to specify only the bare minimum in the standard, rather than defining the system around specific applications, it seemed more judicious to search for the functionalities needed for clusters of applications and to provide the core technology that can fulfill these functionalities. Hence, functionality has become an important issue in the MPEG-4 standardization framework. These observations were reflected in the call for proposals that was issued in 1994, to examine new technologies answering the needs expressed in terms of functionalities. Dozens of proposals in the form of complete algorithms or specific tools were presented and evaluated in November 1995 and January 1996 [9,10]. The first video VM was created based on technologies proposed, in January 1996.

Since then, the experts committee has been continually evaluating new technologies fulfilling existing or new functionalities through core experiments or other equivalent mechanisms. This has led to four versions of the VM at the time of this writing, each either bringing a higher efficiency for an existing functionality, or adding a new functionality to the verification model. The video VM continues to evolve. A video Working Draft for MPEG-4 video is expected in November

1996. This will in turn become the Draft International Standard, by November 1997. The International Standard is expected by November 1998.

1.2 Functionalities addressed

As the MPEG-4 standard intends to support a wide range of multimedia applications, it will certainly support functionalities such as security, low delay, synchronization, interworking, etc. Some of these functionalities have already been or are being addressed by a number of other current or emerging standards. The MPEG-4 standard will use similar or improved solutions in order to address these functionalities under its scope.

Apart from the above, the MPEG-4 committee is seeking solutions for supporting eight key functionalities that are thought not to be well-supported by existing or other emerging standards. These new functionalities have been divided into three major non-orthogonal classes, based on the requirements they support.

- **Content-based interactivity :** This class includes four functionalities focused on requirements for applications involving some sort of interactivity between the user and the data, namely, *content-based multimedia data access tools*, *content based manipulation and bitstream editing*, *hybrid natural and synthetic data coding*, and *improved temporal random access*. Applications benefiting from these functionalities include data retrieval from on-line libraries, interactive home shopping, and movie production and editing.
- **Compression:** This class is composed of two functionalities: *improved coding efficiency* and *coding of multiple concurrent data streams*. They essentially aim at applications requiring an efficient storage or transmission of audio-visual information and their efficient synchronization. These functionalities will enhance some existing applications such as information browsing over Internet, and virtual reality.
- **Universal access:** The remaining two functionalities are: *robustness in error-prone environments* and *content-based scalability*. These functionalities allow MPEG-4 encoded data to be accessible over a wide range of media, and with various qualities in terms of temporal and spatial resolutions for specific objects, which could be decoded by a range of decoders with different complexities. Applications benefiting from these functionalities are wireless communications, database browsing and access at different content levels, scales, resolutions, and qualities.

1.3 Verification model and core experiments

An important phase of the MPEG-4 video standardization is that of development of a VM. A VM is a completely defined encoding as well as decoding algorithm. It is composed of tools and algorithms that serve as a reference to assess the performance of other tools, and algorithms. By definition, a VM should be precise enough to allow multiple independent parties to produce unique and identical results.

So far, the MPEG-4 video group has focused its efforts on a single VM which has gradually evolved from version 1.0 to version 4.0 and, in the process, has addressed an increasing number of desired functionalities, namely, content based object, and temporal scalabilities, frame based spatial scalability, error resilience, and compression efficiency.

The evolution of a VM to new versions is based on *core experiments*. Core experiments are performed to improve existing tools or algorithms in the VM, or to incorporate new ones. Core experiments are proposed by one or more MPEG experts, and approved by consensus, provided that at least two independent experts agree to carry out experiments. Besides the precise definition of the algorithm or tool to be evaluated, a core experiment must also define the parameters and other specifications of the framework under which the core experiment is to be carried out. This would allow a comparison of the results of independent experimenters.

In the past year, dozens of core experiments have been established, evaluating hundreds of new tools and algorithms. Some of the tools and algorithms have been included into the video VM as a result of this process. There are currently eight classes of core experiments in the video MPEG-4 examining tools and algorithms in 43 distinct core experiments. They cover tools and algorithms for prediction, texture coding, entropy coding, rate control, shape and alpha channel coding, object/region texture coding, error resilience, bandwidth and complexity scaling, multi-view coding, model manipulation, as well as pre-, mid-, and post-processing.

In the future, as the video VM reaches more maturity, the number of core experiments and thus the tools and algorithms to evaluate is expected to decrease.

2 GENERAL STRUCTURE OF VERIFICATION MODEL

In the previous section, the objective of MPEG-4 video was described as providing standardized core technologies allowing efficient storage, transmission and manipulation of video data in multimedia environments. This is a challenging task, given the broad spectrum of requirements and applications in multimedia. We also saw that, in order to achieve this goal, functionalities common to clusters of applications are under consideration. Therefore, video activities in MPEG-4 aim at providing solutions in the form of tools and algorithms enabling functionalities such as efficient compression, object scalability, spatial and temporal scalability, and error resilience. The MPEG-4 video standard will provide a set of tools and algorithms bringing solutions to the above mentioned functionalities and more.

To this end, the approach taken in MPEG-4 relies on a content based representation of visual data. In contrast to current state-of-the-art techniques [1,2,7,8], MPEG-4, considers a scene to be a composition of Video Objects (VO) with intrinsic properties such as shape, motion, and texture. It is believed that such a content based representation is key for interactivity with objects for a variety of multimedia applications. In such applications, a user can access arbitrarily shaped objects in the scene and manipulate them.

The process by which a VO is formed (or extracted from a scene) depends on the application and on the exact environment in which the system is used. In the particular case of the video data stream (such as those coded by current standards) a VO is simply the set of rectangular frames of a given size (depending on its format) in the video sequence. In other situations, a VO may represent a 2D or 3D synthetic object generated by a computer which evolves in time. As a third example, one could consider a VO as frames representing a foreground object extracted from a scene by automatic, semi-automatic or supervised segmentation, or even a blue screen process.

The process by which a VO is formed need not be standardized. This is similar to MPEG-2 video standard where, for instance, the process for the computation of the motion vectors in predictive coding is not defined. In contrast, the role of the standard is to provide the representation model of the VO (or the motion vectors in the MPEG-2 example), such that this convention can be used by all decoders in order to decode the VO (or the motion vectors).

Although the process for the definition of the VO is not being standardized, the situation is slightly different for the video VM. By definition, the video VM should be unique and should detail the exact methods of encoding and decoding. Therefore, at least the VO should be clearly defined in the VM. Currently, the following method is used in MPEG-4 video to define the VOs. For all the test sequences used in MPEG-4 video, the VO is defined by either considering the entire video sequence as one single VO, or by providing frames of segmented objects (and their eventual shape or transparency data) each considered as a different VO.

Figure 1 shows the block diagram of the MPEG-4 video encoder. The most important feature of this encoder is the intrinsic representation based on VO when defining a visual scene. In fact, a user, or an intelligent algorithm may choose to encode the different VOs composing a source data with different parameters, different coding methods, or may even choose not to code some of them at all.

In most applications, each VO represents a semantically meaningful object in the scene. In order to maintain a certain compatibility with available video materials, each uncompressed VO is represented as a set of Y, U, and V components plus information about its shape, stored frame after frame in predefined temporal intervals. It is important to note that although in MPEG-4 video test sequences the VO were either known by construction of the sequences (hybrid sequences based on blue screen composition or synthetic sequences) or were defined by semi-automatic segmentation, this is done only due to practical considerations. The same approach remains valid for any objects (synthetic or natural) and any dimensionality (2D or 3D), and even any representation model used per VO.

Another important feature of the video VM is that no temporal frame or frame rate is explicitly defined by this approach. This means that the encoder and decoder can therefore function in different frame rates which do not even need to be constant throughout the video sequence.

Interactivity between user and the encoder or the decoder can be conducted in different ways. The user may decide to interact at the encoding level, either in coding control to distribute, for instance, the available bitrate between different VOs, or to influence the multiplexing to change parameters such as composition script at the encoder. In cases where no back channel is available, or when the compressed bitstream already exists, the user may interact with the decoder by either acting on the compositer to change the position of a VO or its blending order. The user can also influence the decoding at the demultiplexer by requesting the processing of a portion of the bitstream only, such as shape.

Figure 2 shows the block diagram of the decoder in the video VM. The structure of the decoder is basically similar to that of encoder in reverse, except for the composition block at the end. The exact method of composition (blending) of different VOs depends on the application and the method of multiplexing used at the system level. The current VM uses a recursive solution in which blending is performed sequentially with two planes at a time [3].

One important issue that will not be addressed in this paper is that of synchronization among different VOs and other entities such as audio data. The multiplexing and synchronization will be handled by the system level of MPEG-4 standard [11].

2.1 Data structure of the VM

In order to fulfill the required functionalities of the video VM, besides a flexible and powerful representation based on the VO concept, the data structure used in the syntax of encoder/decoder should be designed carefully. The current video VM uses the following classes in its syntax to allow an easy and efficient implementation of functionalities described in section 1.2.

The following hierarchy of classes is used in the video VM syntax:

- VideoSession (VS)

VS is an entity embedding the other three classes. A complete video sequence may be formed of several Video Sessions.

- VideoObject (VO)

VO is a class defining specific objects in a scene. This class reflects the notion of Video Object as described in previous sections. Object scalability is achieved through the use of the VideoObject class.

- VideoObjectLayer (VOL)

VOL is a class that enhances the temporal or spatial resolution of a VO. This class is closely related to notions of spatial and temporal scalabilities.

- VideoObjectPlane (VOP)

VOP is an occurrence of a VO at a given time. Two different VOPs may belong to the same object at two different times and not necessarily to two different objects.

In summary, a Video Session is a collection of one or more Video Objects each of which can consist of one or more layers and each layer consists of an ordered sequence of snapshots in time in the form of VOPs. Thus there can be several VOs (VO0, VO1,..) in a VS and for each VO there can be several layers (VOL0, VOL1,..) and each layer consists of a time sequence of VOPs (VOP0, VOP1,..), which are basically snapshots in time. A VO can be of arbitrary shape (rectangular is a special case). For single layered coding only one VOL (VOL0) exists per VO. Figure 3 shows the hierarchical structure of the syntax.

Currently, due to the lack of 3D/4D data in the MPEG-4 video test data, as well as the lack of appropriate technology for their representation, manipulation and coding, the MPEG-4 video VM is based on Video Object Planes and therefore is limited to the manipulation of 2D objects.

The exact method of multiplexing these different entities depends on the application and coding as well as decoding environments. Multiplexing is described in the MPEG-4 systems working draft.

2.2 VOP based encoding

Figure 4 represents the structure of the VOP encoder. In a given session, the same encoding scheme is applied when coding all the VOPs.

The encoder is composed of two main parts: shape coding and the traditional motion and texture coding applied to the same VOP. Texture coding as well as motion estimation and compensation parts of the encoder are similar in principle to those used in other state-of-the-art standards, where care has been taken in order to extend their respective tools to objects of arbitrary shape. The truly novel component in the coding scheme is that of shape coding which will be described in more detail in the next section.

Although the coding algorithm of the VM is designed to handle arbitrarily shaped objects, all current tools are based on the concept of macroblock. This allows a certain compatibility with other standards and a more straightforward insertion of tools developed for such environments. In order to take advantage of both macroblock and other arbitrarily shaped object coding tools, the multiplexing of the bitstream generated from the coding tools can be made in separate or combined motion-shape-texture modes. In the combined motion-shape-texture mode, the bitstreams of all these features are combined together on a macroblock basis and put in the final

bitstream macroblock after macroblock, per VOP. In separate motion-shape-texture mode, the bitstreams generated for motion, shape or texture of the entire VOP are multiplexed in the final bitstream, each occupying a contiguous portion of the latter.

Moreover, in applications where the shape information is not required (for example a classical rectangular frame based video coding), the shape coding can be disabled.

For very low bitrate applications, where blocking effect may occur, a deblocking filter may be used in the coding loop.

The following section describes the details of some of the major tools used for the coding of the VOPs. Emphasis is given to the coding of the luminance component. Chrominance components are basically treated in a similar way by appropriate subsampling operations except for their motion estimation where the rescaled values of luminance motion vectors are used.

3 CURRENT TOOLS IN THE VIDEO VERIFICATION MODEL

3.1 Shape coding

In the past, the problem of shape representation and coding has been thoroughly investigated in the fields of computer vision, image understanding, image compression and computer graphics. However, this is the first time that a standardization effort is adopting a shape representation and coding technique within its scope.

In the current MPEG-4 VM, two kinds of shape information are considered as inherent characteristics of a video object. These are referred to as binary and grey scale shape information. By binary shape information one means a label information that defines which portions (pixels) of the support of the object belong to the video object at a given time. The binary shape information is most commonly represented as a matrix with the same size as that of the bounding box of a VOP. Every element of the matrix can take one of the two possible values depending on whether the pixel is inside or outside the video object.

The grey scale shape information has a similar corresponding structure to that of binary shape with the difference that every pixel (element of the matrix) can take on a range of values (usually 0 to 255) representing the degree of the transparency of that pixel. The grey scale shape corresponds to the notion of alpha plane used in computer graphics, in which 0 corresponds to a completely transparent pixel and 255 to a completely opaque pixel. Intermediate values of the pixel correspond to intermediate degrees of transparencies of that pixel. By convention, a binary shape information corresponds to a grey scale shape information with values of 0 and 255.

In its canonical form, a binary or grey scale shape is represented as a matrix of binary or grey scale values called bitmap or alpha plane. However, for the purpose of compression, manipulation, or a more semantic description, one may choose to represent the shape in other forms such as using geometric representations or by means of its contours.

Since its beginning, the video VM has adopted a bitmap based compression technique for the shape information. This is mainly due to the relative simplicity and higher maturity of such techniques. Core experiments have shown that bitmap based techniques offer good compression efficiency with relatively low computational complexity. In future developments of the video VM or MPEG-4 coding toolbox, based on the results of other core experiments, it is possible that at least one alternative shape coding technique based on representation models allowing a more semantic description finds its way into the final standard.

This section describes the coding methods for binary and grey scale shape information. Binary shape information is encoded by a motion compensated block based technique allowing both lossless and lossy coding of such data. Grey scale shape information is encoded using a block based motion compensated DCT similar to that of texture coding, allowing lossy coding only. The grey scale shape coding also makes use of binary shape coding for coding of its support. In the current VM, the shape of every VOP is coded along with its other characteristics. To this end, the shape of a VOP is bounded by a rectangular window with a size of multiples of 16 pixels in horizontal and vertical directions. The position of the bounding rectangle is chosen such that it

contains the minimum number of blocks of size 16x16 with non transparent pixels. The samples in the bounding box and outside of the VOP are set to 0 (transparent). The rectangular bounding box is then partitioned into blocks of 16x16 samples (hereafter referred to as shape blocks) and the encoding/decoding process is performed block by block.

3.1.1 Binary shape coding

The binary shape coding technique used in the VM is a motion compensated block based technique applied to consecutive VOPs belonging to the same VO. Figure 5 shows an example of a VOP illustrating its bounding box and showing how it is partitioned into shape blocks of size 16x16 as the first step for its encoding process.

Each shape block is then coded by detecting its color-changing pixels (from opaque 255 to transparent 0, and vice versa), and by calculating the distances between successive changing pixels. If all the pixels in a shape block are of the same color, the coding is not carried out. In this case only a flag is transmitted to the decoder informing it that the shape information for that shape block is either all transparent or all opaque. Figure 6 shows an example of the binary shape coding for a given shape block illustrating the changing pixels positions.

The distance between consecutive changing pixels are coded as a result of this algorithm. Three modes are used in order to represent this distance, namely, horizontal mode, vertical pass mode, and vertical mode. Each mode corresponds to a different way of representing the distance between changing pixels inside the shape block and their respective binary codewords [3].

The binary shape information can be size converted (subsampled) if a lossy mode is to be used for the binary shape coding. The current down conversion ratios are 4:1 and 16:1, respectively.

The influence of a lossy shape coding on the total bitrate cannot be estimated directly, as a lossy shape influences the amount of the bits required to represent the texture and motion for a lossy reconstructed VOP. This is an important issue to be aware of when designing rate control mechanisms at the encoding stage. The influence of the lossiness of the shape on the motion and the texture coding parts of the VM is currently under investigation.

3.1.2 Binary shape coding with Motion Compensation(MC)

In order to improve the compression efficiency of the binary shape coding, the motion vectors of texture macroblocks in a VOP are used to reduce the temporal redundancy of consecutive VOP shapes. In this case, the shape blocks are either intra- or inter-coded with motion compensation. No residual error is coded in inter-coding of binary shape information.

The motion compensation is applied to binary shape with the same scheme as for luminance macroblocks, except that no padding and overlapping operations are performed. The selection of intra- or inter-coding depends on the shape prediction error and on the coding mode (intra/inter) of the texture macroblocks at the same spatial position of the shape block.

3.1.3 Grey scale shape coding

In this technique, the grey scale shape information is coded by separately processing its support as a binary shape, and the transparency value of its pixels as luminance values. The support function is encoded by the same binary shape coding technique that was described above. The transparency values are encoded as luminance macroblocks of size 16x16 with the same technique which is used for the coding of the texture information of VOPs. Figure 7 illustrates the principle behind grey scale shape coding method used in the video VM.

3.2 Motion estimation and compensation

Motion estimation and compensation have been introduced in the video VM in order to exploit the temporal redundancies existing in most video sequences. The principle behind temporal redundancy reduction in the MPEG-4 video VM is similar to those used in other standards [1, 2, 7, 8].

The main differences arise from the fact that these motion estimation and compensation techniques need to be extended to deal with arbitrarily shaped objects which are represented as VOPs. This has become possible by dividing the arbitrarily shaped VOPs into macroblocks of size 16x16, and by means of two new tools called padding and modified block (polygon) matching motion estimation. Every VOP can be coded in three possible ways, namely, I-VOP, P-VOP and B-VOP, depending on whether the VOP is coded completely independently (Intra, I-VOP), or it is predicted from a previous VOP (Predictive, P-VOP), or interpolated from a past and a future VOP (Bi-directional interpolation, B-VOP). Figure 8 illustrates an example of each of these coding modes.

The motion estimation is performed by first dividing the bounding box of the VOP into macroblocks of size 16x16 pixels. For macroblocks completely outside of the VOP but inside its bounding box, no motion estimation is performed (see Figure 9). For macroblocks entirely inside a VOP, the motion estimation is carried out in a manner similar to other standards by means of block matching for macroblocks of size 16x16 as well as for their respective 4 subblocks of size 8x8 pixels, resulting in one motion vector per macroblock and one for each of the 4 subblocks, with a half sample accuracy. The reference pixels used for the motion estimation are those of a previous or future decoded I-VOP or P-VOP. Various decision modes are used in order to choose between the inter- or intra-coding for each macroblock as well as for the number of motion vectors assigned to it.

The motion estimation for macroblocks only partially inside the VOP is performed using a modified block (polygon) matching technique. In this technique, the matching error is computed as the sum of absolute value differences formed by those pixels of the current macroblock that are inside of the VOP shape, and that of the corresponding pixels in the reference VOP (see Figure 9). As some of the reference pixels used in the matching may be outside of the reference VOP, a block based repetitive padding is performed in order to extrapolate the value of these pixels from those inside of the reference VOP. A similar padding scheme is used to extend the reference VOP beyond its bounding box. This improves efficiency, by allowing more possibilities when searching

for candidate pixels for prediction at the boundary of the reference VOP. This latter procedure is known as unrestricted motion vector search [3,8].

When a VOP is inter-coded, the motion vectors for its prediction have to be transmitted in the bitstream to the decoder. The motion vector components (horizontal and vertical) are differentially coded by predicting its value from up to three neighboring motion vectors already transmitted. The codeword used to code the differential motion vector data depends upon the range of the vectors when performing the motion estimation. This maximum range is determined at the encoder and transmitted to the decoder in a way similar to the one used in the MPEG-2 standard [2].

In order to improve the prediction quality, a technique called overlapped motion compensation is used in the video VM. This approach is similar to the overlapped motion compensation used in H.263 technique. For each block within a macroblock to be predicted, the motion vector from immediately above, below, left or right, as well as the one associated to the current block are used in order to obtain up to five estimates of the current block from the reference VOP. These estimates are then in turn averaged according to predefined weighting factors in order to obtain the final result for the predicted current block.

3.3 Texture coding

Texture corresponds to the pixel values in the case of an intra-coded VOP (I-VOP) or to the residual error after the prediction in the case of an inter-coded VOP (P-VOP or B-VOP). The texture coding technique used in the video VM is similar in several aspects to those used in other state-of-the-art standards [1,2,7,8]. Due to its simplicity and rather good performance, and also in order to incorporate in a straightforward manner functionalities such as error resilience into the current VM, the texture coding of the video VM is chosen to be a block based technique, where special care is taken in order to extend the block based approach to handle arbitrarily shaped VOP coding.

Again, the bounding box of an intra-coded VOP or its corresponding motion compensated residual error is split into a number of macroblocks of size 16x16.

The intra VOP macroblocks and the residual macroblocks after motion compensation are coded one after the other using a Discrete Cosine Transform (DCT) scheme. DCT is performed separately on each of the luminance and chrominance planes in a given macroblock, totaling 6 blocks of size 8x8. Similarly to motion estimation and compensation, three types of macroblocks may be encountered in a VOP bounding box: those that lie completely inside the VOP shape; those completely outside of the VOP but inside the bounding box; and those that partially cover the VOP. The macroblocks that are completely outside of the VOP are not coded at all. Those macroblocks that lie completely inside the VOP are coded using a conventional DCT scheme. The 8x8 blocks of macroblocks lying partially on the VOP are first padded using repetitive padding as for the motion estimation, with the difference that for residue blocks, the region outside of the VOP within the blocks are padded with zero values. If all the pixels in an 8x8 block are transparent, their values are replaced by zero. These blocks are then coded in a manner identical to the blocks inside of the VOP. Figure 10 illustrates typical cases of texture coding in an arbitrary shape context.

The DCT coefficients are quantized, zig-zag scanned, and entropy coded by run-length Huffman method. The quantization step can be based on one of the two following alternatives. The first, similar to that of recommendation H.263, uses a quantization parameter that is exploited to quantize the AC coefficients in the DCT. This value may change based on a desired quality or a targeted bitrate, on a macroblock by macroblock basis. The second alternative for quantization is similar to that used in MPEG-2 where the quantization step size may vary depending on the position of the AC coefficient in the frequency domain according to a quantization matrix. The default quantization matrix used is flat with a value of 16 except for the DC value. This quantization matrix can be overwritten by downloading different intra or non-intra quantization matrices. In all cases the DC coefficients are quantized with a step size equal to 8.

For a higher compression efficiency, the DC as well as the first row and first column of the AC coefficients in the intra-coded macroblocks can be differentially coded using an adaptive predictive method. The prediction value used is chosen as the corresponding value of the block above or immediately to the left of the current block. This adaptive selection is based on comparison of horizontal and vertical gradients of corresponding DC values of already coded neighboring blocks.

3.4 Scalable coding

Content based spatial and temporal scalability capabilities are two important functionalities required by a number of applications. The video VM therefore supports both spatial and temporal scalabilities using the VOL structure. The following describes how such functionalities can be achieved.

A bitstream is scalable if at least one subset of the bitstream is sufficient for generating a useful presentation. Scalable video coding involves generating a coded representation in a manner which facilitates the derivation of video of more than one resolution/quality by scalable decoding. Bitstream scalability is the property of a bitstream that allows decoding of appropriate subsets of a bitstream to generate complete pictures of resolution and/or quality commensurate with the proportion of the bitstream decoded. If a bitstream is truly scalable, decoders of different complexities, from low to high performance, can coexist and while low performance decoders may decode only small portions of the bitstream producing basic quality, high performance decoders may decode much more and produce significantly higher quality results from the same bitstream.

Each type of scalability involves more than one VOL. In the following, we limit ourselves to two layers only. In the case of two layers consisting of a lower layer and a higher layer, the lower layer is referred to as the base-layer and the higher layer is called the enhancement-layer. Traditionally, these scalabilities are applied to frames of video such that in the case of spatial scalability, the enhancement-layer frames enhance the spatial resolution of base-layer frames, while in temporal scalability, the enhancement-layer frames are temporally multiplexed with the base-layer frames to provide video with higher temporal resolution. Many MPEG-4 applications are however even more demanding and necessitate not only traditional frame based scalabilities but also scalabilities of VOPs of arbitrary shapes.

The scalability framework of the video VM is referred to as generalized scalability and includes spatial and the temporal scalabilities. In the case of temporal scalability, the VM supports both frames (rectangular VOPs) as well as arbitrary shaped VOPs. However, in the case of spatial

scalability, only rectangular VOPs are presently supported. Figure 11 shows a generalized codec structure for the two-layer scalability in the video VM.

VOPs are input to a scalability pre-processor. If spatial scalability is to be performed with the base-layer at lower spatial resolution and the enhancement-layer at higher spatial resolution, the pre-processor performs spatial downsampling of the input VOPs to generate a first base-layer which forms the input to the MPEG-4 VOP encoder in Figure 4. The reconstructed VOPs from the base-layer are then fed to the mid-processor which in this case performs a spatial upsampling. The other output of the pre-processor corresponds to the higher spatial layer VOPs and forms the input to the MPEG-4 enhancement-layer encoder which is similar in structure but different in strategy to that of the base-layer encoder. The method for the encoding of the enhancement-layer is described below. The base- and enhancement-layer bitstreams are multiplexed by the MPEG-4 system multiplexer. The decoder is similar to the encoder in the reverse order with the difference that the scalability post-processor performs any necessary operations such as spatial upsampling of the decoded base-layer for display.

When the generalized codec is used to introduce temporal scalability, the scalability pre-processor performs temporal demultiplexing of a VO into two substreams of VOPs, one of which is input to the MPEG-4 base-layer encoder and the other to the MPEG-4 enhancement-layer encoder. In this case, mid-processor does not perform any spatial resolution conversion and simply allows the decoded base-layer VOPs to pass through. These VOPs are used for temporal prediction while encoding of enhancement-layer. The operations of MPEG-4 system multiplexer and demultiplexer are exactly the same as in case of spatial scalability. The decoding of base- and enhancement-layer bitstreams occurs in the corresponding base- and enhancement-layer decoders as shown in Figure 11. At the decoder, the post-processor simply outputs the base-layer VOPs without any conversion, but temporally multiplexes the base and enhancement-layer VOPs to produce a higher temporal resolution enhancement-layer.

The video VM allows two types of enhancement mechanisms:

- **Enhancement type 1:** The enhancement-layer increases the resolution of a portion of the base-layer.
- **Enhancement type 2:** The enhancement-layer increases the resolution of the entire base-layer.

Figure 12 illustrates an example of a scene containing several regions, which is coded as a base-layer with lower spatial or temporal resolution. In the case of an enhancement of type 1, the car is enhanced in either spatial or temporal resolution. In this case, the entire frame has been coded as as one VO in VOL0 and the car only as enhancement-layer VOL1. In the case of an enhancement of type 2, either the entire scene or the car is enhanced in temporal or spatial resolution and coded as a VOL1 layer, depending on whether the VOL0 represents the entire frame or the car only, in its base-layer.

3.4.1 Spatial Scalability

As mentioned earlier, for spatial scalability, the base-layer and the enhancement-layer have different spatial resolutions. If needed, a downsampling process is performed by the scalability pre-processor. The VOP in the base-layer is encoded with exactly the same technique as described for

the non-scalable case in the previous sections. The VOP in the enhancement-layer is encoded as either P-VOP or B-VOP. The relationship between the VOP in the base-layer and that of the enhancement-layer is illustrated in Figure 13. The VOP that is temporally coincident with an I-VOP in the base-layer is encoded as a P-VOP. The VOP that is temporally coincident with a P-VOP in the base-layer is encoded as a B-VOP. In case of the spatial scalability, a decoded VOP in the base-layer is used as a reference for the prediction, after upsampling. The temporally coincident VOP in the reference layer (base-layer) must be coded before the encoding of the VOP in the enhancement-layer.

3.4.2 Temporal Scalability

In temporal scalability, the temporal resolution of a selected object is enhanced such that it has a smoother motion than the remaining area.

Figure 14 shows the example of a temporal scalability of enhancement type 1 where VOL0 is an entire frame with both an object and a background, while VOL1 represents a particular object in VOL0. VOL0 is coded with a low frame rate, while VOL1 is coded to achieve a higher temporal resolution. In this example, frames 2 and 4 are predicted from the base-layer frame 0 followed by overlapping the object of the enhancement-layer onto the combined frames 0 and 6. The combined frame is formed using background composition [3]. In this case, forward prediction is used to form P-VOPs. Figure 15 shows another example of temporal scalability of enhancement type 1 which uses bi-directional predictions to form B-VOPs in the enhancement-layer.

Figure 16 shows an example of a temporal scalability with enhancement type 2, where VO0 represents a sequence of a rectangular background without any enhancement-layer. Moreover, VO1 represents a sequence of a particular object coded with two layers, VOL0 and VOL1. VOL1 represents the same object as VOL0 and is coded as an enhancement-layer to achieve a higher temporal resolution with respect to the base-layer VOL0. Note that the VO0 may not have the same frame rate as other VOs.

3.5 Error robustness

A very important feature of the video VM is its robustness to errors in adverse environments such as wireless networks. Robustness is achieved by inserting resynchronisation marker-fields in the bitstream with approximately constant spacing. The resynchronisation marker-field should be inserted by the encoder before the first macroblock, and also if the number of output bits since the last resynchonisation exceeds a predetermined value.

The insertion of synchronization markers for error robustness can be disabled when such functionality is not needed. When enabled, the recommended configuration of the VM is in the combined shape-motion-texture coding, with the AD/DC prediction in intra block texture coding disabled.

4 CONCLUSIONS

This paper gives an overview of the MPEG-4 Video VM version 4.0. It is important to note that, by the time of publication of this paper, other versions of the MPEG-4 Video VM will have been issued. These versions will be different from the one described in this paper (version 4.0) for some of the tools but not for its basic approach.

The architecture of the VM and related data structures allowing efficient coding and manipulation of arbitrary shape content are described. In particular, the coding algorithms and major tools included in the video VM are discussed. Currently, work is in progress, through core experiments, to improve the algorithms and associated tools in the VM. In addition, new tools are foreseen that will add new functionalities. Current core experiments in the MPEG-4 have been rather focused on applications requiring low to medium quality images. However, work has started to extend this scope to higher quality applications, as a certain maturity is being reached in the former.

Concrete results show that MPEG-4 video coding based on a content representation architecture and providing novel functionalities such as content-based interactivity, is now a reality. Only the future will tell the impact that MPEG-4 standard will have in the emergence of the multimedia applications. However what has been achieved so far, and the increasing interest on the activities of the MPEG-4 standardization are very encouraging signals that this impact will be very positive.

Acknowledgments

The author would like to acknowledge all the MPEG-4 members, especially those in the video and verification model development for their insights, fruitful collaborations and inputs which led to the definition of the video VM, and also this paper. Without their collaborative efforts, as well as their commitment and endless work, the MPEG-4 would have not been as it is today. The author also wishes to especially thank Fernando Pereira and Sushil Bhattacharjee for careful reading of this paper and their valuable comments and suggestions.

Biography

Touradj Ebrahimi was born on July 30, 1965. He received his M.Sc. and Ph.D., both in Electrical Engineering, from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 1989 and 1992 respectively. From 1989 to 1992, he was a research assistant at the Signal Processing Laboratory of the Swiss Federal Institute of Technology (EPFL). During the summer 1990, he was a visiting researcher at the Signal and Image Processing Institute of the University of Southern California, Los Angeles, California. In 1993, he was a research engineer at the Corporate Research Laboratories of Sony Corporation in Tokyo, where he conducted research on advanced video compression techniques for storage applications. In 1994, he served as a research consultant at AT&T Bell Laboratories working on very low bitrate video coding. He is currently at the Signal Processing Laboratory of EPFL, where he is involved with various aspects of Digital Video and Multimedia applications and in charge of the Digital TV group. In 1989, he was the recipient of the IEEE and Swiss national ASE award. He is also the head of Swiss delegation to MPEG and JPEG and represents the Swiss national body at ISO/IEC JTC1/SC29. His research interests are multidimensional signal processing, image processing, information theory, and coding. He is the author or the co-author of more than 70 research publications, and 6 patents.

Dr. Ebrahimi has been involved in MPEG-4 standardization activities since its beginning. He is currently acting as the chair of the ad-hoc group on video VM editing.

5 REFERENCES

[1] ISO/IEC JTC1 CD11172, « Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s - Part 2: Coding of Moving Picture Information », International Organization for Standardization, 1991.

[2] ISO/IEC DIS 13818-2, « Information Technology - Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video », International Organization for Standardization, 1994.

[3] MPEG Video Group, «MPEG-4 Video Verification Model Version 4.0 ». ISO/IEC JTC1/SC29/WG11/M1380, Chicago meeting, October 1996.

[4] R. Schaeffer and T. Sikora, « Digital Video Coding Standards and Their Role in Video Communications », Proceedings of the IEEE, Vol. 83, No. 6, June 1995.

[5] L. Chiariglione, « The development of an integrated audiovisual coding standard: MPEG », Proceedings of the IEEE, Vol. 83, No. 2, February 1995.

[6] F. Pereira, « MPEG-4: a New Challenge for the Representation of Audio-Visual Information », Proceedings of Picture Coding Symposium'96, Melbourne, March 1996.

[7] CCITT SG XV, « Recommendation H.261 - Video Codec for Audiovisual Services at px64 kbit/s », COM XV-R37-E, International Telecommunication Union, August 1990.

[8] ITU-T Draft, « Recommendation H.263 - Video Coding for low bitrate communication », International Telecommunication Union, November 1995.

[9] T. Alpert, V. Baroncini, D. Choi, L. Contin, R. Koenen, F. Pereira and H. Peterson, « Subjective Evaluation of MPEG-4 Video Codec Proposals: Methodological Approach and Test Procedures », in this issue.

[10] J. Ostermann, « Methodologies Used for Evaluation of Video Tools and Algorithms in MPEG-4 », in this issue.

[11] O. Avaro, *et al.* « The MPEG-4 Systems and Description Languages: A Way ahead in Audio Visual Information Representation », in this issue.

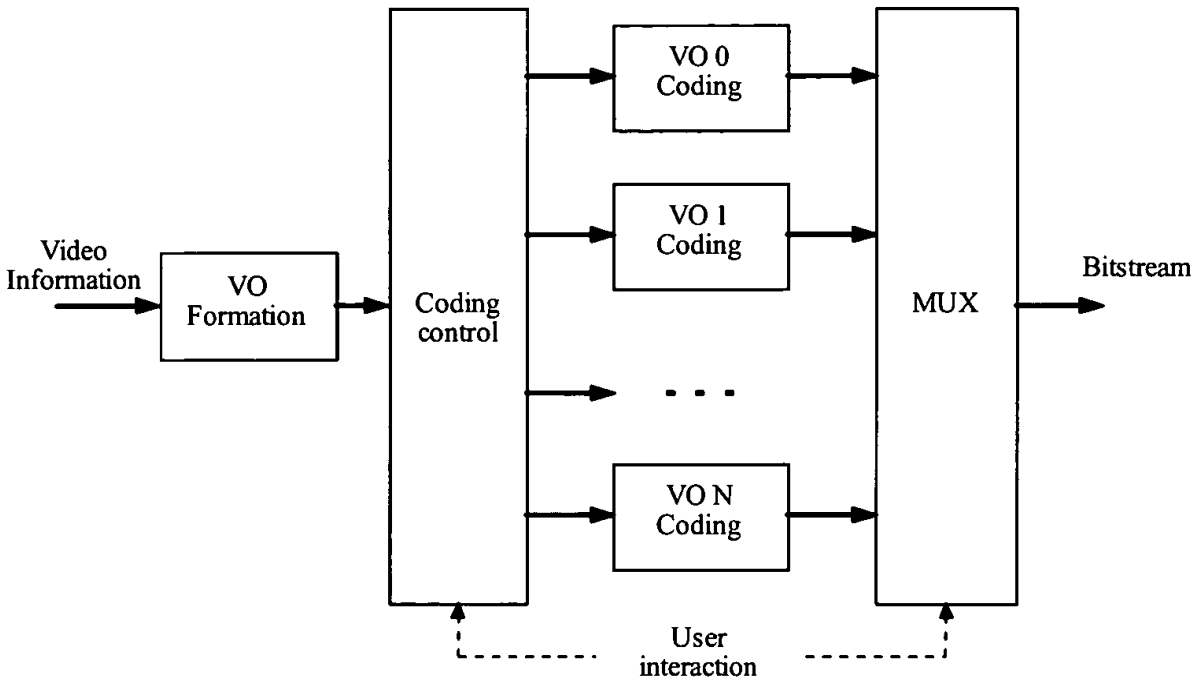

Figure 1: General structure of the encoder in the MPEG-4 video VM.

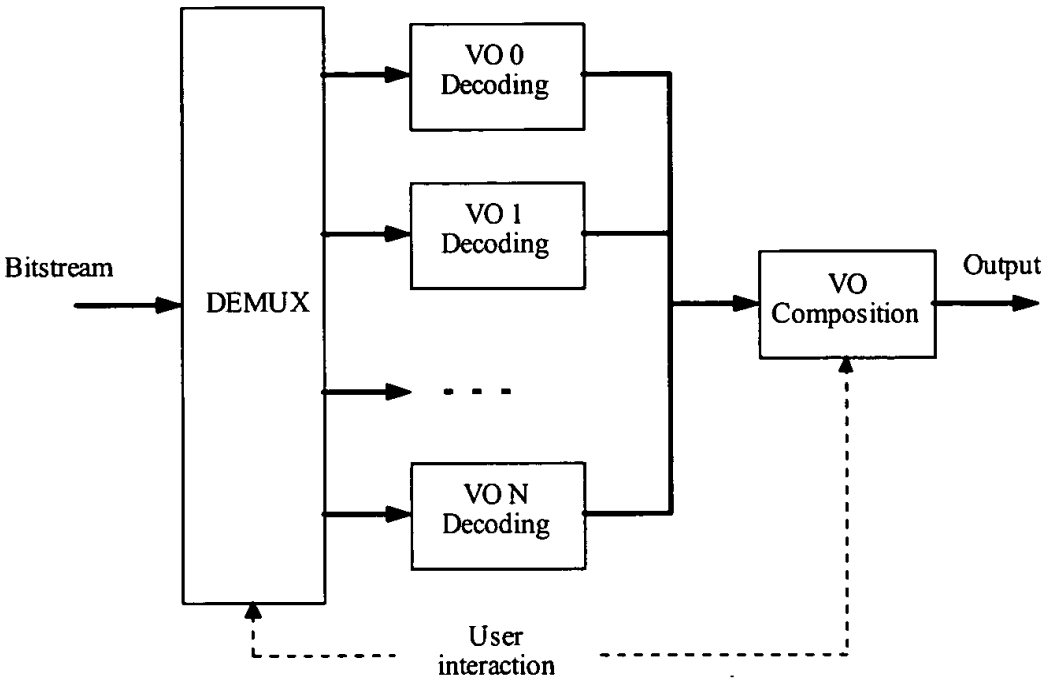

Figure 2: General structure of the decoder in MPEG-4 video VM.

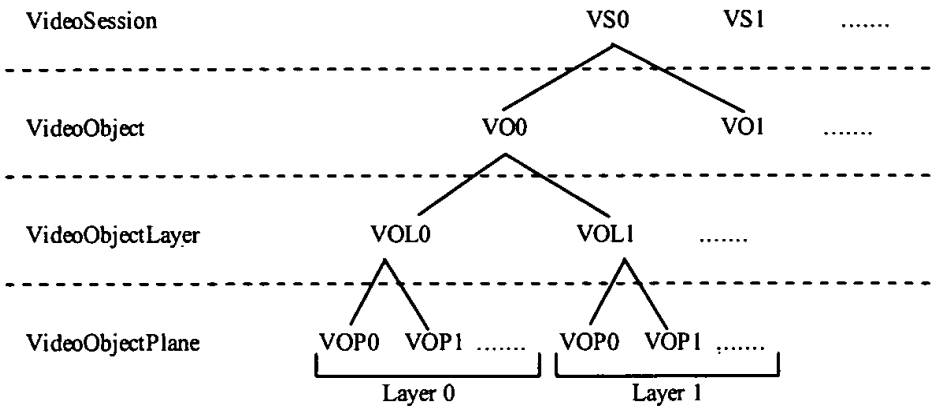

Figure 3: Hierarchy of the data structure classes in the MPEG-4 video VM.

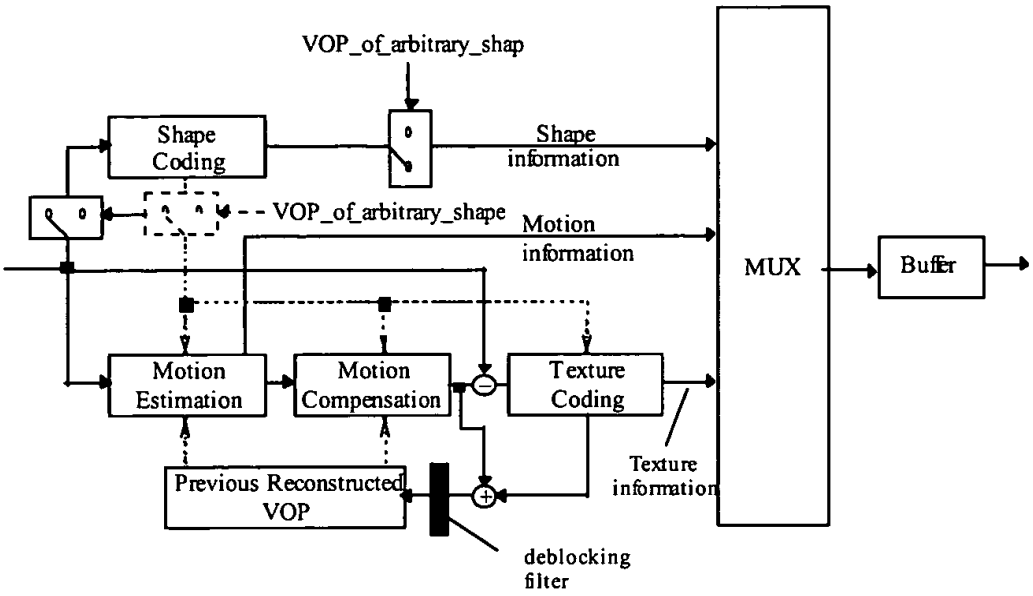

Figure 4: Video Object Plane based encoder structure in the MPEG-4 video VM.

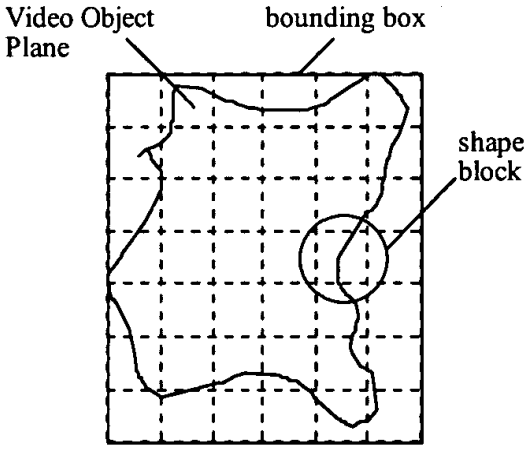

Figure 5: Shape coding procedure of a Video Object Plane.

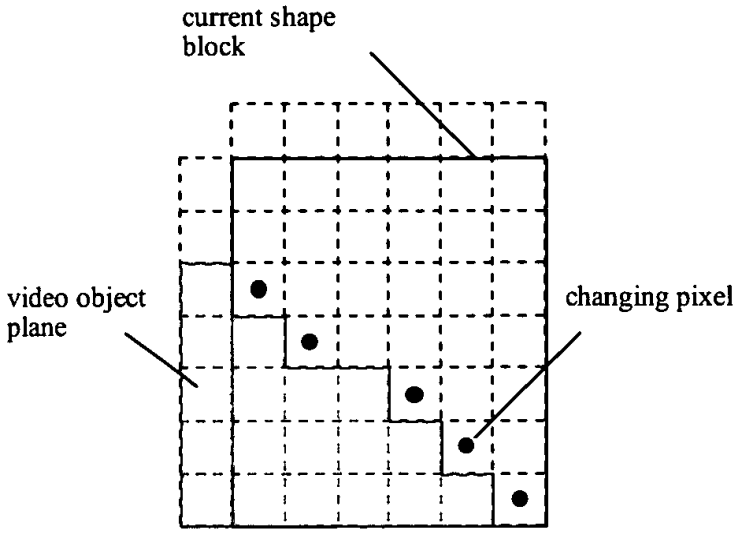

Figure 6: Binary coding of a shape block, based on changing pixels.

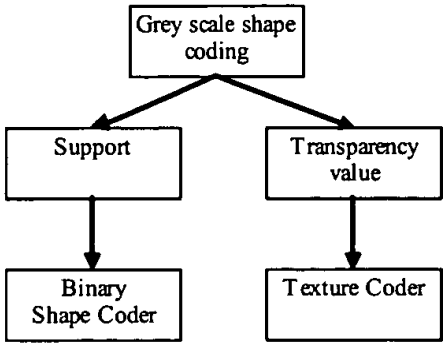

Figure 7: Grey shape coding technique used in the video VM.

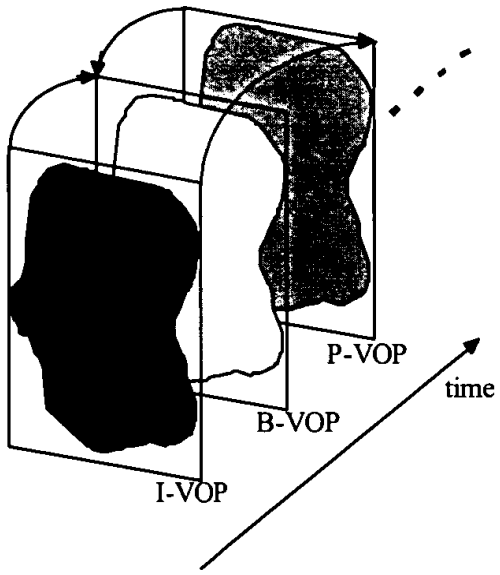

Figure 8: VOP coding modes in the video VM.

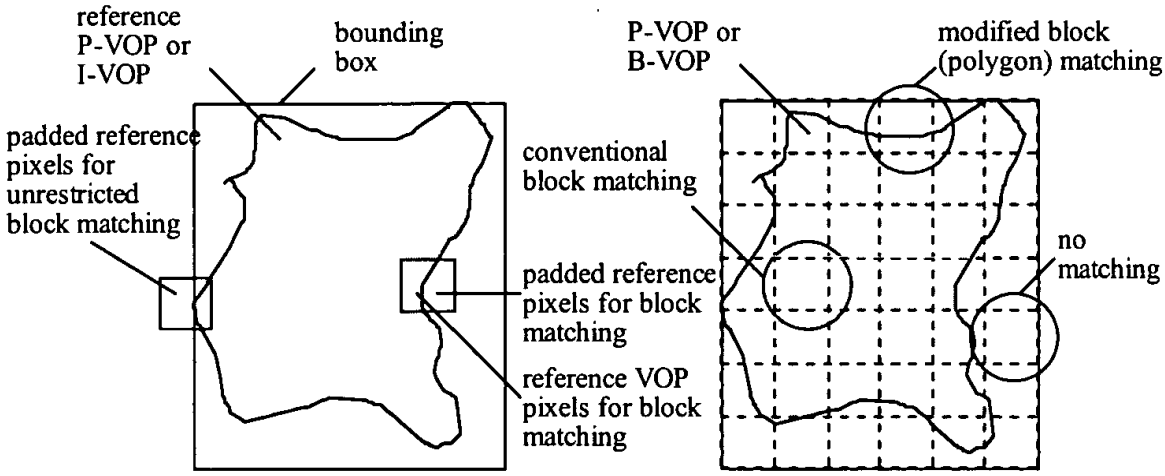

Figure 9: Motion estimation for arbitrarily shaped VOPs.

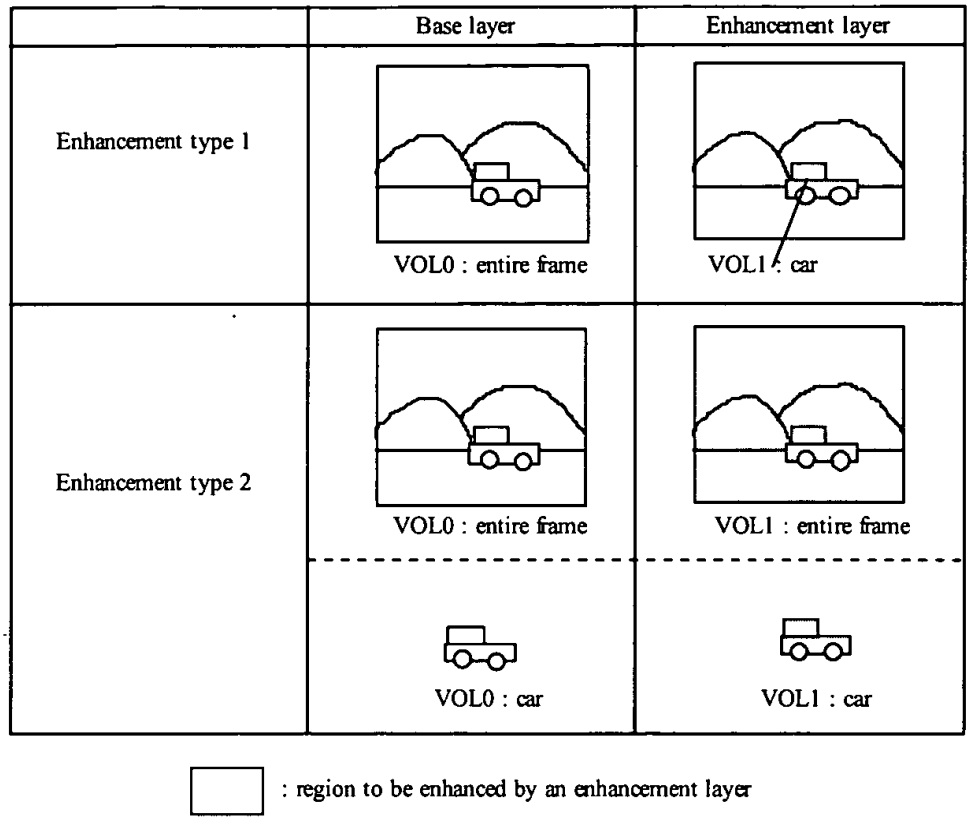

Figure 12: Example of enhancement types in scalable coding.

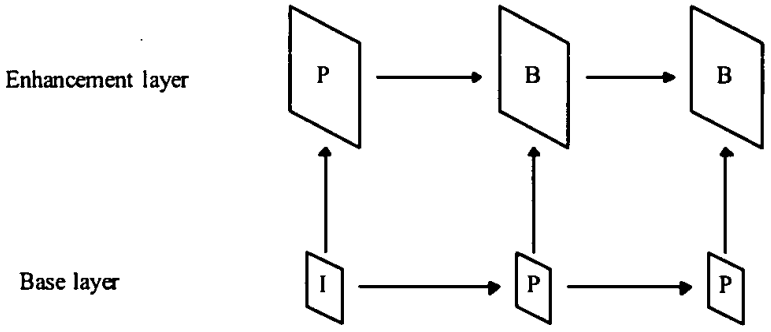

Figure 13: Coding of the enhanced layer for spatial scalability.

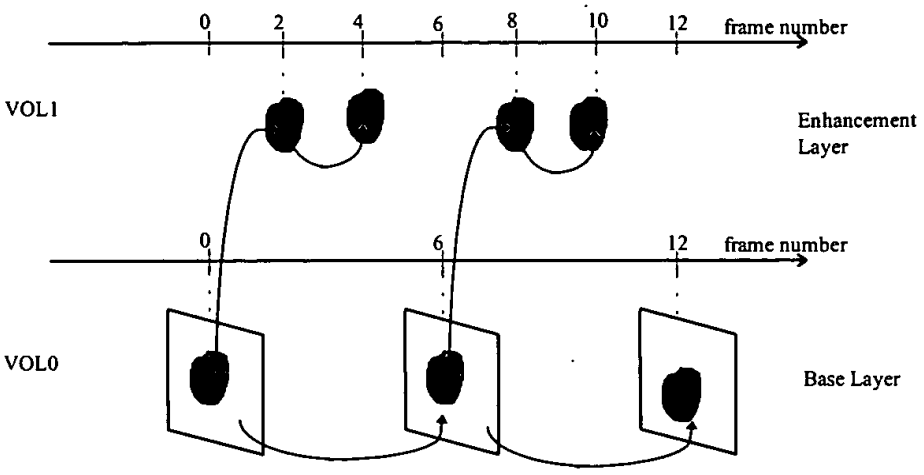

Figure 14: Temporal scalability of enhancement type 1 with P-VOPs.

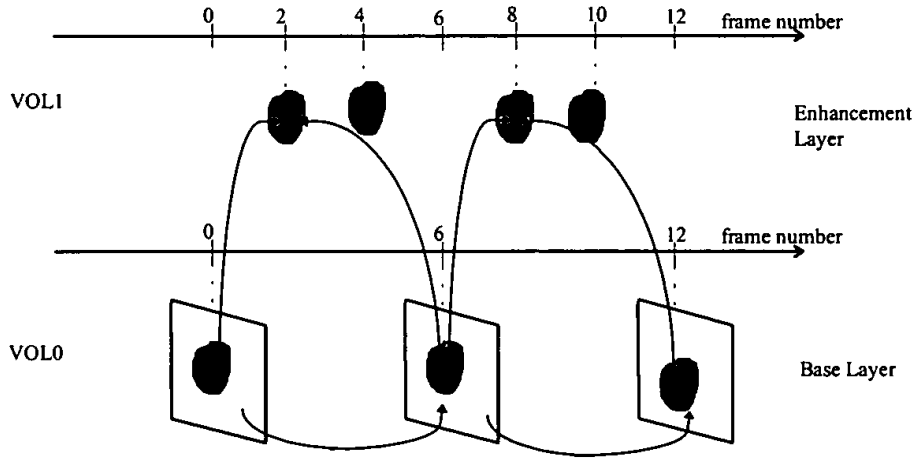

Figure 15: Temporal scalability of enhancement type 1 with B-VOPs.

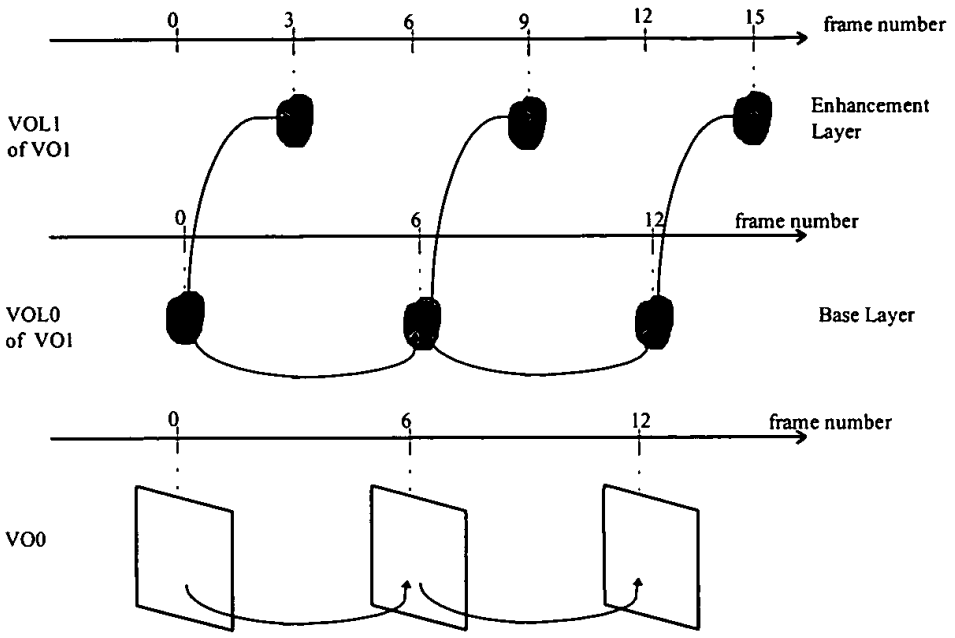

Figure 16: Temporal scalability of enhancement type 2.